MarCO: Interplanetary Mission Development on a CubeSat Scale

Josh Schoolcraft, Andrew Klesh and Thomas Werne
Jet Propulsion Laboratory, California Institute of Technology, California, USA

I. Introduction

SHORTLY after JPL's Interior Exploration using Seismic Investigations, Geodesy and Heat Transport (InSight) mission launches, separates, and commences its cruise phase, two CubeSats will deploy from the launch vehicle's upper stage and begin independent flight to Mars (Fig. 1). During InSight's entry, descent, and landing (EDL) sequence, these twin Mars Cube One (MarCO) spacecraft will fly 3,500 km above the Martian surface, recording and relaying InSight UHF radio data to the Deep Space Network (DSN) on Earth[1].

MarCO is a twin CubeSat mission developed by the NASA Jet Propulsion Laboratory (JPL) to accompany the InSight (Interior Exploration using Seismic Investigations, Geodesy and Heat Transport) Mars mission lander. MarCO's primary mission objective is to launch with InSight and independently fly to Mars to serve as a communications relay during InSight's entry, descent, and landing (EDL) phase. MarCO represents a new type of deep space mission: CubeSats at Mars. Building on the development of JPL's first interplanetary CubeSat project, the Interplanetary Nano-Spacecraft Pathfinder in Relevant Environment (INSPIRE), MarCO further refined the approach to hardware, software, and ground architecture development to solve the challenges of quickly building low-budget spacecraft to fly to Mars. The greatest constraint, beyond others typical of CubeSat missions, was time. The duration between MarCO's conception to completion of spacecraft assembly was less than two years - an unprecedented schedule for any planetary mission to date. Through necessity, MarCO has built on previous experience, procedures, systems, and development methodologies, defining a new niche for supporting larger primary missions. The MarCO spacecraft are poised to write a new chapter in deep space exploration.

Originally slated to launch and reach Mars in 2016, the InSight mission schedule subsequently slipped to 2018. During the original landing of InSight, Earth would not be in view, and no orbiter around Mars would have been in position to both receive UHF EDL data and simultaneously relay it back to Earth. It was from this obstacle that MarCO was conceived. Regardless of any changes to InSight's 2018 EDL configuration geometry, MarCO is still expected to fly and serve in the same capacity as originally designed: the first CubeSat mission to Mars.

CubeSats have historically been firmly in the domain of universities and small companies. As first conceived, they served as a platform upon which to teach all aspects of the space mission lifecycle. JPL took on this mission type with Interplanetary Nano-Spacecraft Pathfinder in Relevant Environment[2] (INSPIRE), moving the concept into a new domain: deep space. Building from the INSPIRE platform and lessons learned, MarCO addressed new challenges in the domain of planetary missions: independent interplanetary flight and navigation, integration with a large-scale mission, long-distance and long-delay communication, short development time, and a small development team. Of these, the greatest constraint was schedule: only 18 months passed from conception of mission concept until delivery of fully assembled and tested flight hardware. Careful selection of mission team, along with extensive use of off-the-shelf equipment, and streamlining automated processes, was essential. This achievement represents the next step in the evolution of CubeSats beyond low-Earth orbit.

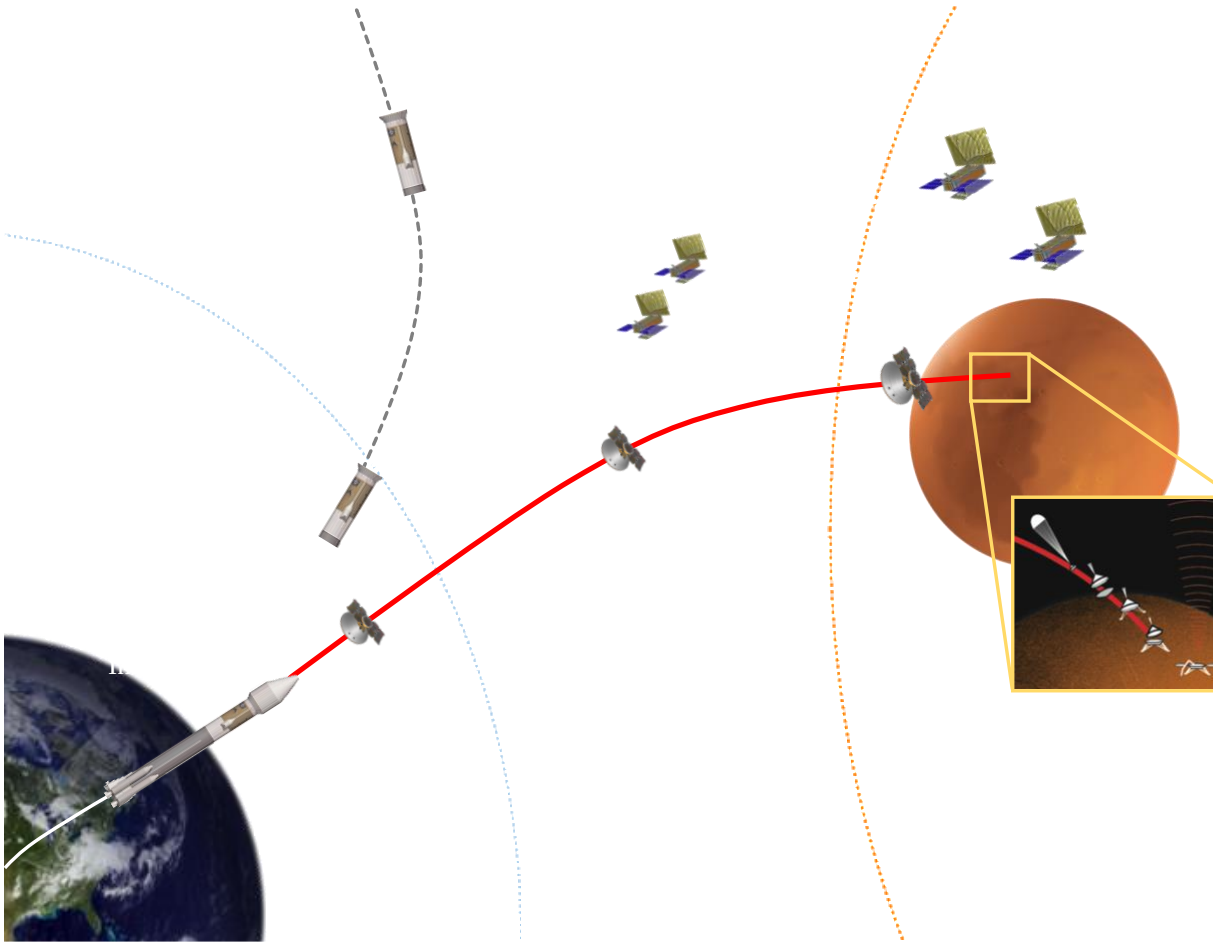

Figure 1. Basic MarCO concept of operations. *Trajectories for both MarCO spacecraft (A and B) are shown in white; the InSight trajectory in shown in red. 4 total trajectory correction maneuvers (TCMs) will occur for each MarCO spacecraft during the 6.5-month cruise phase prior to InSight's EDL.*

II. Mission Status

Development and assembly is complete. Hardware for both spacecraft has been received, tested, and integrated into the spacecraft stack. In late 2015, InSight's launch was postponed. Despite this schedule disruption, MarCO remains ready for the opportunity to launch to Mars along with InSight. After this launch, MarCO will conduct operations as originally designed.

III. Spacecraft Hardware

Each MarCO spacecraft is 6U in size (1U = 10x10x10 cm). They have several deployable elements: solar arrays, high-gain X-band reflectarray antenna as shown in Fig. 2, and a UHF antenna, seen in Fig. 3. Additionally, each spacecraft has a command and data handling system (C&DH), electrical power system (EPS), two cameras, an attitude determination and control system (ADCS), and at the heart of the mission, the V2 Iris radio.

A. C&DH Subsystem (Fig. 2)

Provided by AstroDev LLC, the MarCO C&DH is an incremental development based on a previous design used by the INSPIRE project[1]. It consists of an MSP430F2618 flight microcontroller (FCPU), an MSP430FR5739 functioning as a real-time clock (RTC), 48 megabytes of nonvolatile phase-change memory, and a cascaded watchdog system. The FCPU handles interfaces (SPI, I^2C, UART/RS-422, and standard GPIO) to all spacecraft subsystems, effectively routing relevant data to and from appropriate devices.

Designed by the MarCO team, the RTC is in reality a general-purpose microcontroller integrated into the C&DH subsystem. The RTC executes custom software to provide "RTC-plus-plus" functionality accessible on the I^2C spacecraft bus. In addition to being a clock, MarCO's RTC also acts as a I^2C-UART bus communications bridge for transferring data to and from each camera system, and provides general nonvolatile storage in its ferromagnetic RAM (FRAM). Because FRAM read/write performance and endurance rivals that of SRAM[3], the RTC microcontroller is able to save frequently updated variables (such as the current time and telemetry sequence counters) even between full power cycles of the entire spacecraft bus.

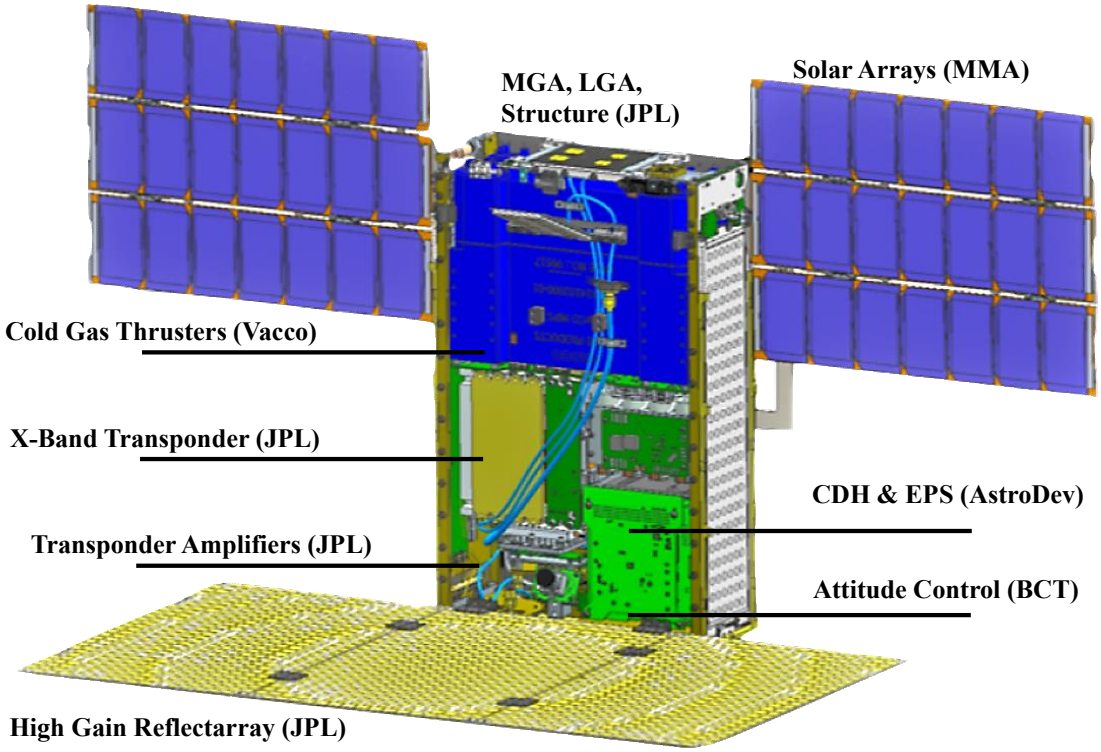

Figure 2. Front of MarCO spacecraft. *Cutaway reveals cold gas thruster propellant tank, Iris X-Band transponder / amplifiers, C&DH / EPS, and ADCS system.*

B. Electrical Power System (EPS) and Solar Arrays (Fig. 2)

Supplied by AstroDev LLC, the EPS subsystem allows four channels of solar panel input, and regulates 12V battery, 5V and 3.3V buses. MarCO's battery pack consists of 18650S lithium ion cells in 3S4P configuration. MarCO's solar arrays were acquired from MMA Design LLC. Through a 4-motion deployment process, the dual arrays unfold from narrow 3U panels (launch position) and rotate into the fixed flight configuration shown in Figs. 2 and 3.

C. Attitude Control System (ACS) and Propulsion (Fig. 2)

MarCO's XACT attitude control system (Fig. 2) is provided by Blue Canyon Technologies (BCT). It includes a star tracker, gyro, sun sensors, and reaction wheels. It has modifications to software and hardware to accommodate deep-space operations. Cold-gas propulsion functionality comes via an 8-valve and tank system from Vacco Industries (Fig. 2), and is directly controlled by the XACT unit. Both systems are also controllable by the C&DH subsystem, although during normal operations, minimum interaction beyond power control should be necessary.

D. Spacecraft Structure, Harnessing, and Thermal (Fig. 3)

All structure, cabling, and interface boards were developed at JPL, and maintain compliance with the 6U CubeSat standard[4]. Thermal mitigation is achieved with two radiators, blankets, and heaters, in conjunction with numerous temperature sensors throughout each spacecraft bus.

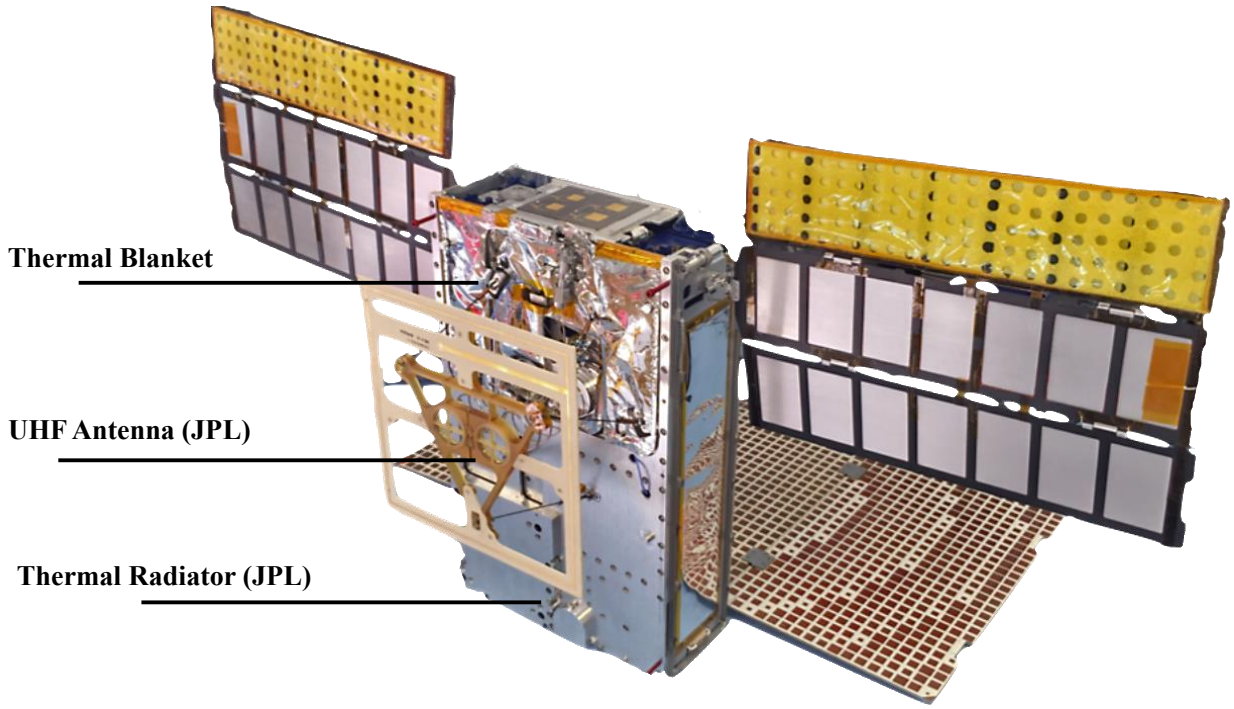

Figure 3. Rear of MarCO spacecraft, via mechanical fit check assembly. *All external flight-like hardware assembled to evaluate pre-deployment stowage.*

E. Cameras

Each of the two fixed-zoom 752 x 480 pixel resolution camera is connected to an independent GumStix single-board computer for image capture and processing. One camera uses a narrow field-of-view lens (NFOV), while the other uses a wide field-of-view (WFOV) lens. The entire assembly remains unpowered until the C&DH enables each system. The NFOV camera points down along the UHF antenna boresight, facing Mars during the EDL flyby. The WFOV camera faces the HGA, allowing for both verification of reflectarray & feed deployment, as well as imaging of the Earth. After capture, each image is processed into several thumbnails and compressed to various levels prior to downlink. As previously noted, the RTC microcontroller bridges the camera communications with the FCPU. When ready for downlink, the images are packetized by the C&DH, and forwarded to the radio.

F. Radio and Antennas

Like the C&DH subsystem, the Deep Space Network (DSN)-compatible software-defined Iris radio brings heritage from the INSPIRE project. Supporting 4W X-band transmission output, MarCO's V2 Iris radio was customized to include a UHF receiver, on-board Consultative Committee for Space Data Systems (CCSDS) protocol processing, radiation tolerance, external solid state power amplifier (SSPA), and external low-noise amplifier (LNA).

Each spacecraft has four antennas:

1) Wide beam width, low-gain patch antenna (LGA) for near-Earth communications (Fig. 2);
2) Medium gain patch array (MGA) for safe-mode communications far from Earth (Fig. 2);
3) High gain reflectarray antenna (HGA) for 8kbps data at 1.05AU (Fig. 2); and
4) UHF loop antenna to receive data from the InSight lander during EDL (Fig. 3).

IV. Mission Software

MarCO utilizes custom flight software developed by a two-person software team. While most of the code runs in the FCPU, there are additional pieces in the RTC microcontroller, the watchdog microcontroller, the camera systems, and the V2 Iris radio's CCSDS protocol library.

On the ground, the MarCO project extensively developed to, tested with, and plans to fly using JPL'S AMPCS telemetry and command processing software tool.

G. Flight Software

1. Core Flight Software

MarCO's core flight software, named "protos"[5], has pieces of heritage from a number of small JPL projects and missions, generally resource-constrained. The current instantiation of protos (Fig. 4) was originally developed by the same software team for INSPIRE, and adapted to MarCO. While the INSPIRE and MarCO C&DH FCPU (a 24-bit Von Neumann architecture MSP430F2618 from Texas Instruments) has extensive flight heritage in CubeSat missions, it is extremely resource-limited: 8 kilobytes of SRAM, 116 kilobytes of flash program memory, running at 12 MHz. Careful resource conservation was, and remains, essential. To support both the processing and schedule constraints of the mission, flight software development occurred in a very tight loop, almost entirely on engineering hardware. Due to the relatively low costs associated with CubeSat systems, a C&DH system was always available for development, allowing high fidelity testing. While adhering to a prudent level of code review and schedule oversight, the software team was given wide latitude to make and change design decisions as necessary to ensure mission success. The flight code is optimized for size, leveraging automation in the code-generation and build processes as needed. The final flight software consists of approximately 20,000 lines of handwritten C plus another 5,000 lines of auto-generated C parameter tables. Compiled, this takes roughly 85% of FCPU internal flash memory. Because this utilization is above 50%, entirely new software releases cannot be uploaded in flight. However, the capability exists - and has been successfully tested - to patch and extend executing code in real-time through a series of uplink commands.

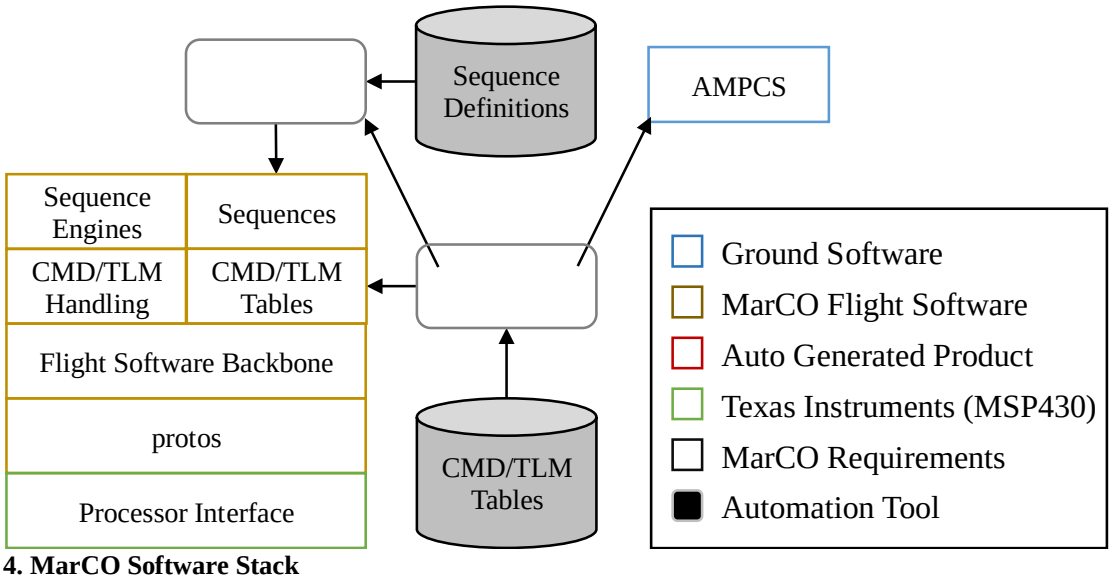

Figure 4. MarCO Software Stack

Despite running on a radiation "soft" processor, the flight code is written with reliability in mind. Critical variables and tables are stored with triple modular redundancy (TMR), pointers are repeatedly verified, and code execution order is enforced. Failures due to SRAM corruption result in a software reset, which takes less than one second to complete. The code is robust to resets, capable of restoring critical state values as necessary. Table-driven tasks, such as command and telemetry dictionary distillation, generate both flight and ground software at build time, essentially eliminating the possibility of introducing errors in translating from dictionaries to raw code.

The core task that the MarCO flight software accomplishes is sequence execution. Any spacecraft task can be accomplished through the use of a sequence. Sequences are written in a low-level programming language created by

the software team for the MarCO mission. The MarCO sequencing language supports standard programming constructs such as variables, conditional branching, and looping. Additionally, it supports the execution of telecommands. Sequences are written in a human-readable format similar to assembly and compiled before being written into the spacecraft FCPU flash memory. Sequences can be created, patched, and deleted in a manner similar to flight software patching. To execute, a sequence is loaded into a sequence engine, which executes the compiled instructions. MarCO supports the execution of up to eight simultaneous sequences, although only three are needed for day-to-day operations.

2. Other Flight Software

The RTC processor is an MSP430FR5739 microcontroller running at 24MHz, with an additional timekeeping crystal oscillator. The RTC-FCPU bus interface is I^2C, and the RTC acts as an I^2C slave peripheral. Like the FCPU, it executes an adaptation of the protos software, with a focus on four primary tasks: timekeeping, GPIO manipulation, camera communications, and FCPU data storage/retrieval (Fig. 5). In the timekeeping capacity, the RTC produces a pulse-per-second signal that can be read throughout the spacecraft bus. It constantly saves current time (using TMR) to a protected area of FRAM at 4Hz, allowing reliable recovery even in the event of a full spacecraft reset. To support camera communications, the RTC controller functions as a buffered I^2C-UART bus bridge between the FCPU and the camera systems. Finally, it acts as generic non-volatile memory device, allowing the FCPU to read and write spacecraft state data that may be lost otherwise during a system reset.

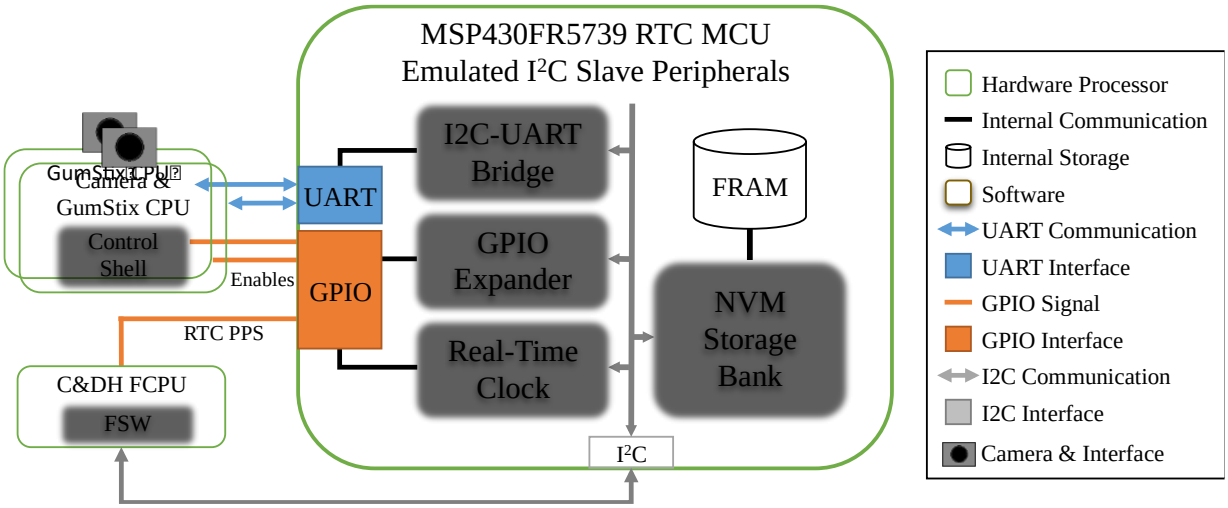

Figure 5. MSP430FR5739 Microcontroller

MarCO's camera systems, two per spacecraft, run a customized embedded Linux distribution on a GumStix single-board computer (Fig. 5). A Python-based control shell spawns at boot on each system, and effectively serves as the interface between camera and spacecraft. The script receives commands from and sends data to the FCPU via the RTC microcontroller, which acts as an I^2C-UART bridge. A simple communications packet protocol defines the way the camera and FCPU interact, and includes basic data integrity checking through the use of CRC16-CCITT. The camera systems are powered on for short periods of time directly by FCPU GPIO lines to perform brief imaging campaigns, process the images, and store them on standard SD cards. Images can be retrieved any time the systems are powered.

Processing downlink telemetry packets into CCSDS Advanced Orbiting Systems (AOS)[6] version 2 space data link protocol telemetry frames is accomplished in the V2 Iris radio with a software library written by the MarCO team. The library, written in portable generic C, supports frame encapsulation of CCSDS Space Packets and allows multiple methods of multiplexing AOS frame virtual channels into a telemetry stream. It also includes a Python abstraction layer and has been tested and used on a wide range of platforms, from desktop PCs (Windows, OSX and Linux) to other embedded architectures. The MarCO project also used it in an Arduino configured as a V2 Iris radio emulator, allowing flight system testing independent of the actual radio subsystem.

H. Ground Software

During the development of Mars Science Laboratory (MSL) at JPL, a tool was created to simplify the real-time processing, storage, and retrieval of mission test data[7]. Written in Java, the AMMOS Multimission Data Processing

and Control System (AMPCS) software package takes advantage of modern software interfaces to give developers, spacecraft testers, and mission operators a flexible system through which to monitor and control spacecraft systems. Fully capable of interfacing with a full range of data streams, from simple serial up to the DSN, AMPCS is a 'Swiss army knife' in the toolbox of low-cost mission development at JPL.

A key factor in the decision to use AMPCS for INSPIRE and MarCO was the ability to start utilizing it early – from initial spacecraft development to full flight operations. Figure 6 shows the high-level dataflow through AMPCS. Any number of testing and flight configurations can be setup simply by providing spacecraft connectivity to the "Telemetry I/O" interface. This true-to-form "test as you fly" capability made AMPCS an ideal choice for both missions.

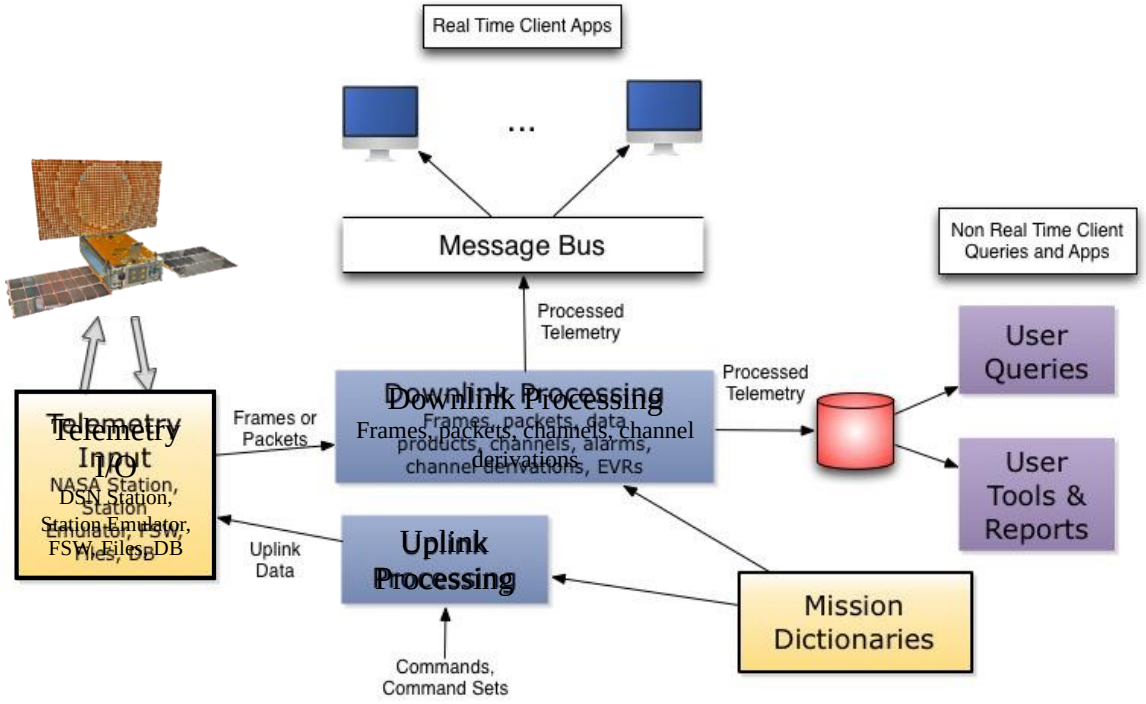

Figure 6. AMPCS dataflow block diagram[7].

AMPCS provides multiple ways to receive, process, and store CCSDS telemetry data throughout a mission lifecycle. The telemetry can be viewed through a standard or mission-customized set of fixed GUI views, and also retrieved through a set of command-line tools for analysis in any other desired fashion. Uplink commanding is similarly supported in a separate module, giving users the ability to build CCSDS telecommands directly from a mission-defined command dictionary and transmit to the spacecraft through a variety of connectivity options.

Deployment of AMPCS for MarCO was simple: virtual machines containing pre-configured instances of AMPCS were instantiated at the necessary locations. Automated build scripts then performed the final software changes as needed based on testing circumstances. Test personnel consolidated testing session data into a primary database for archival and later retrieval.

Additionally, AMPCS incorporates a Python framework for generalized automation. Python scripts can be used to perform functions such as monitoring telemetry for a specific condition, and automatically sending commands when the condition is met. This is particularly useful in testing situations where a common set of commands are used for regression testing of a flight software release.

V. Conclusions

MarCO represents an evolution in the role of CubeSats from low-Earth-orbit teaching and technology demonstration platforms to legitimate deep space and planetary missions. As such, MarCO addressed new challenges, including independent interplanetary flight and navigation, integration with a much larger primary mission, long-distance and long-delay communications, short development time, and a small development team. As planetary missions follow the same trend in technological advancement of improved capability for lower mass and

volume, interplanetary CubeSats will become increasingly important. MarCO is the trailblazer of this new class of planetary missions.

Acknowledgments

This work has been carried out at the Jet Propulsion laboratory, California Institute of Technology, under contract to NASA. Government sponsorship acknowledged.

References

[1] Klesh, Andrew T., Krajewski, Joel A., "MarCO: CubeSats to Mars in 2016," *29th Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, Aug 8-13, 2015.

[2] Klesh, Andrew T., et al. "INSPIRE: Interplanetary NanoSpacecraft Pathfinder in Relevant Environment." *AIAA SPACE 2013 Conference & Exposition*, San Diego, California, Sep 10-12, 2013.

[3] Thanigai, P., Goh, W., "MSP430 FRAM Quality and Reliability," Texas Instruments, SLAA526A, May 2014.

[4] Hevner, R., Holmans, W., Puig-Suari, J., Twiggs, Robert T., "Payload for Canisterized Satellite Dispenser (CSD) Specification Sheet," PSC, 2002206 Rev A Payload Specification for 6U, 12U and 27U, Jun 13, 2011.

[5] Schoolcraft, J., "Back to Basics: Flight Software in 8 Kilobytes," *2014 Workshop on Flight Software*, Pasadena, California, Dec 16-18, 2014.

[6] "AOS Space Data Link Protocol," CCSDS 732.0-B-3, Sep 2015.

[7] Quach, William L., DeForrest, Lloyd R., Klesh, Andrew T., Schoolcraft, Joshua B., "Adapting a Large-Scale Multi-Mission Ground System for Low-Cost CubeSats," *SpaceOps 2014 Conference*, Pasadena, California, May 5-9, 2014.